The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows:

1. A method of editing data having a fixed format, comprising the steps of:
   (a) receiving a first data byte array;
   (b) determining the encoding of the data byte array by determining a number of bytes in each of a plurality of fixed-length fields that comprise a fixed-length statement;
   (c) determining a number of bytes in the fixed-length statement;
   (d) creating a first data string from the first data byte array, given a starting byte position and the number of bytes in the fixed-length statement; and
   (e) assigning an attribute to each byte of the first data string.

2. The method of claim 1, further comprising the step of repairing an ending of the first data string.

3. The method of claim 2, wherein the step of repairing the end of the first data string comprises the steps of:
   (a) determining if the last byte of the first data string is a second byte of a double byte character, and if so, setting a value and changing the attribute of a second last byte to be a shift-out character and removing the last byte;
   (b) if not, then determining if the last byte of the first data string is a first byte of a double byte character and if so, then setting the value and changing the attribute of the last byte to be a shift-out character;
   (c) if not, determining if the last byte of the first data string is a shift-out character, and if so, then removing the last byte of the first data string;
   (d) if not determining if the last byte of the first data string is a shift-in character and if so, determining if a second last byte is a shift-out character, and if so, removing the last two bytes of the first data string.

4. The method of claim 1, further comprising the step of repairing a beginning of the first data string.

5. The method of claim 4, wherein the step of repairing the beginning of the first data string comprises the steps of:

(a) determining if the first byte of the first data string is a second byte of a double byte character, and if so, setting a value and changing the attribute of a first byte to be a shift-out character;

(b) if not, determining if the first byte of the first data string is a first byte of a double byte character and if so, then setting the value and changing the attribute of the second byte to be a shift-out character, and removing the first byte;

(c) if not, determining if the first byte of the first data string is a shift-in character, and if so, then removing the first byte of the first data string;

(d) if not determining if the first byte of the first data string is a shift-out character and if so, determining if a second byte is a shift-in character, and if so, removing the first two bytes of the first data string.

6. The method of claim 1, further comprising the steps of:

(a) determining if the first data string is less than the fixed-length;

(b) if so, then appending spaces to the end of the first data string so the first data string is left-aligned.

7. The method of claim 1, further comprising the steps of:

(a) determine if the first data string is less than the fixed-length;

(b) if so, then prepending spaces to the beginning of the first data string so the first data string is right-aligned.

8. The method of claim 1, further comprising:

(a) expanding the first data string for editing.

9. The method of claim 8, further comprising:
   (a) editing the first data string.

10. The method of claim 8, wherein the step of expanding the first data string comprises the steps of:
   (a) making a copy of the first data string;
   (b) for each byte that has an attribute of shift-out, insert a space;
   (c) for each byte that has an attribute of shift-in, insert a space;
   (d) represent each single byte character as a Unicode equivalent;
   (e) represent each double byte character as a Unicode equivalent;
   (f) construct a byte array with the above substitutions.

11. The method of claim 8, wherein said step of expanding the first data array comprises the steps of:
   (a) making a copy of the first data string;
   (b) for each byte that has an attribute of shift-out, insert a parser-recognized shift-out character;
   (c) for each byte that has an attribute of shift-in, insert a parser-recognized shift-in character;
   (d) represent each single byte character as its respective Unicode equivalent;
   (e) represent each double byte character as its respective second Unicode equivalent and a copy of the respective second Unicode equivalent; and
   (f) construct a byte array with the above substitutions.

12. The method of claim 9, further comprising the steps of:
   (a) returning the edited first data string; and
   (b) recreating a Unicode string from the edited first data string.

13. The method of claim 9, further comprising the steps of:

(a) returning the edited first data string; and

(b) recreating a byte array of fixed-format in EBCDIC.

14. A method of editing data having a fixed-length code, comprising the steps of:

(a) receiving a first data byte array;

(b) determining the encoding of the data byte array by determining a number of bytes in each of a plurality of fixed-length fields that comprise a fixed-length statement;

(c) determining a number of bytes in the fixed-length statement;

(d) creating a first data string from the first data byte array, given a starting byte position and the number of bytes in the fixed-length statement;

(e) assigning an attribute to each byte of the first data string;

(f) creating a plurality of subsets of the first data string; each of the subsets corresponding to a fixed-length field,

(g) repairing a end of each of the plurality of subsets if necessary by

(i) determining if the last byte of the subset is a second byte of a double byte character, and if so, setting a value and changing the attribute of a second last byte of the subset to be a shift-out character and removing the last byte;

(ii) if not, then determining if the last byte of the subset is a first byte of a double byte character and if so, then setting the value and changing the attribute of the last byte to be a shift-out character;

(iii) if not, determining if the last byte of the subset is a shift-out character, and if so, then removing the last byte of the subset;

(iv) if not determining if the last byte of the subset is a shift-in character and if so, determining if a second last byte is a shift-out character, and if so, removing the last two bytes of the subset;

(h) repairing a beginning of the each of the subsets if necessary by:

(i) determining if the first byte of the subset is a second byte of a double

byte character, and if so, setting a value and changing the attribute of a first byte to be a shift-out character;

(ii) if not, determining if the first byte of the subset is a first byte of a double byte character and if so, then setting the value and changing the attribute of the second byte to be a shift-out character, and removing the first byte;

(iii) if not, determining if the first byte of the subset is a shift-in character, and if so, then removing the first byte of the subset;

(iv) if not determining if the first byte of the subset is a shift-out character and if so, determining if a second byte is a shift-in character, and if so, removing the first two bytes of the subset;

(i) determining if the subset is less than the fixed-length, and if so, determining if the subset is to be left-aligned or right-aligned;

(j) if the subset is to be left-aligned, appending spaces to the end of the subset;

(k) if the subset is to be right aligned, prepending spaces to the beginning of the subset; and

(l) combining the subsets into a second data string; and

(m) expanding the second data string for editing.

15. A computer system, comprising:

(a) a first central processing unit (CPU) connected to a first computer memory storing data in a fixed-length format;

(b) a CPU connected to a second computer memory storing data in a format other than the fixed-length format;

(c) an object-oriented class in one of either the first CPU or the second CPU, the object- oriented class comprising:

(i) a Unicode string of data;

(ii) a code page encoding specification;

(iii) a byte array of the data from the Unicode string of data;

(iv) a plurality of attributes, one attribute assigned to each byte of the byte array; and

(v) a plurality of methods that operate on the byte array.

16. The computer system of claim 15, wherein the code page encoding specification is EBCDIC.

17. The computer system of claim 15, wherein the code page encoding specification is ASCII.

18. The computer system of claim 15, wherein the plurality of methods further comprise:

(a) a method to get the Unicode String method and a method to get a byte array length from the code page encoding specification.

19. The computer system of claim 15, further comprising a first constructor method to input a Unicode string and output a byte array in the fixed-length code page encoding specification.

20. The computer system of claim 15, further comprising a second constructor method to create a Unicode string from a byte array.

21. The computer system of claim 15, further comprising a method to create a subset array of the byte array.

22. The computer system of claim 21, further comprising a method to truncate the subset array to the fixed-length.

23. The computer system of claim 22, further comprising a method to repair the beginning and/or the end of the subset array.

24. The computer system of claim 15, further comprising a method to right-align and/or left- align the byte array.

25. The computer system of claim 15, further comprising a method to expand the Unicode string into an editable byte array.

26. A computer system for the transfer of data, comprising:

(a) application means to read an original string of data not having a fixed-length format;

(b) means to input a coding specification having a fixed-length format;

(c) means to create a substring of the original string of data having a fixed-length;

(d) means to truncate the substring;

(e) means to repair the beginning and/or the end of the truncated substring;

(f) means to right-align or left-align the repaired truncated substring;

(g) means to expand the substring; and

(h) means to edit the substring.

(i) means to convert the edited substring to Unicode receive in a second computer; and

(j) means to read and decode said and adaptively reconstruct said data based.

27. The computer system of claim 26, further comprising:

(a) means to convert the edited substring to a data code format having a fixed-length format.

28. The system of claim 26, further comprising:

(a) means to convert the edited substring to a data code format not having a fixed-length format.

29. A medium for transmission of an application to be implemented on a processing device, the application comprising the machine-implementable steps of:

(a) receiving a Unicode data string:

(b) creating a substring from the Unicode data string; the substring having a fixed-length format;

(c) assigning attributes to each byte of the Unicode data string, an attribute indicating if a Unicode character is a single byte or a double byte character;

(d) truncating the substring;

(e) repairing the beginning and/or end of the substring; and

(f) creating a expandable form of the substring.